Solving the Distributed Two Machine Flow-shop Scheduling Problem using Differential Evolution

Paul Dempster[1], Penghao Li[2], and John H. Drake[3]

[1] Artificial Intelligence and Optimisation Research Group,
School of Computer Science, University of Nottingham Ningbo China,
Ningbo, 315100, China
paul.dempster@nottingham.edu.cn
[2] School of Computer Science, University of Nottingham Ningbo China,
Ningbo, 315100, China
zy10611@nottingham.edu.cn
[3] Operational Research Group, Queen Mary University of London,
Mile End Road, London, E1 4NS, UK
j.drake@qmul.ac.uk

Abstract. Flow-shop scheduling covers a class of widely studied optimisation problem which focus on optimally sequencing a set of jobs to be processed on a set of machines according to a given set of constraints. Recently, greater research attention has been given to distributed variants of this problem. Here we concentrate on the distributed two machine flow-shop scheduling problem (DTMFSP), a special case of classic two machine flow-shop scheduling, with the overall goal of minimising makespan. We apply Differential Evolution to solve the DTMFSP, presenting new best-known results for some benchmark instances from the literature. A comparison to previous approaches from the literature based on the Harmony Search algorithm is also given.

1 Introduction

Flow-shop scheduling is a classic optimisation problem with an extensive literature dedicated to studying a number of different variants. At its core, the aim is to generate a schedule of jobs to be completed on a set of machines, where each job has a given processing time on each machine, optimising for a particular objective. Although some variants of the problem can be solved easily, minor extensions can lead to a great increase in computational complexity. When the processing times for each job are fixed, although the two-machine case is solvable by a polynomial time algorithm [4], it becomes NP-hard when a third machine [3] or operation [5] is added.

Differential Evolution (DE) [9] is a popular population-based metaheuristic which has been particularly successful at solving continuous optimisation problems [1]. Despite the fact that DE is chiefly designed to operate in a continuous environment, it has frequently been adapted to solve combinatorial optimisation problems and in particular, flow-shop scheduling problems. Onwubolu and

Davendra [6] defined methods to transform solutions between real-valued vector and integer permutation representations, searching in continuous space whilst evaluating solution quality in discrete space. Qian et al. [8] combined differential evolution operating in continuous space with local search in discrete space to solve a multi-objective variant of the flow-shop scheduling problem with buffers in-between machines. Other work by Pan et al. [7] and Wang et al. [11] defined specific operators in discrete space to perform direct search on flow-shop problems using DE.

Here we focus on the distributed two machine flow-shop scheduling problem (DTMFSP). This variant of flow-shop scheduling considers a distributed set of locations containing two machines, with a set of jobs of varying lengths which must be processed on both machines at a single location. Previous work by Deng et al. [2] compared three variants of the Harmony Search algorithm over a set of benchmarks to this problem. In this paper we apply DE to the benchmark instances for the DTMFSP and compare to the previously presented Harmony Search based approaches.

2 The Distributed Two Machine Flow-shop Scheduling Problem (DTMFSP)

The well-known two machine flow-shop scheduling problem [4] is defined as follows. Given a set of n jobs $\mathcal{J} \in \{J_1, ..., J_n\}$, to be processed on two machines M_1 and M_2. Each job $J_i \in \mathcal{J}$ consists of two sequential operations O_{i,M_1} and O_{i,M_2} with associated processing time p_{i,M_1} and p_{i,M_2} respectively. A schedule is a permutation $\pi = (\pi(1), ..., \pi(n))$ of jobs representing the order in which the jobs are processed on the two machines. Under the constraints that each machine can only process one job at a time and jobs must be processed sequentially on the two machines, using makespan as the performance criteria, the goal is to minimise the overall time taken to complete all of the jobs. This problem can be solved to optimality in polynomial-time using Johnson's rule [4].

The distributed two machine flow-shop scheduling problem (DTMFSP) is an extended version of this problem [2]. The difference between the two is that jobs in the DTMFSP can be processed in any of f identical factories, each containing a machine $M_{f,1}$ and a machine $M_{f,2}$, whereas the classical problem considers only a single factory. Additionally, jobs cannot be transferred to another factory, i.e. once a job has been processed on machine $M_{f,1}$ at a given factory, it is not possible to then process the second operation of the job on a machine in another factory $M_{g,2}$ where $g \neq f$. Rather than searching directly on the space of possible job permutations, when solving the DTMDSP search takes place over a vector of integers representing the allocated factory of each job. For each set of schedules, the overall makespan can be calculated as the longest completion time of a single factory after applying Johnson's rule to each factory.

Given a processing sequence π^k at a particular factory k, with a total of n_k jobs at each factory, the makespan of a set of schedules C_{MAX} can be calculated

as:

$$C_{\pi^k(1),M_1} = p_{\pi^k(1),M_1}, k = 1, 2, ..., f \tag{1}$$

$$C_{\pi^k(i),M_1} = C_{\pi^k(i-1),M_1} + p_{\pi^k(i),M_1}, k = 1, 2, ..., f; i = 2, 3, ..., n_k \tag{2}$$

$$C_{\pi^k(1),M_2} = C_{\pi^k(1),M_1} + p_{\pi^k(1),M_2}, k = 1, 2, ..., f; \tag{3}$$

$$C_{\pi^k(i),M_2} = max\{C_{\pi^k(i),M_1}, C_{\pi^k(i-1),M_2}\} + p_{\pi^k(i),M_2}, k = 1, 2, ..., f; i = 2, 3, ..., n_k \tag{4}$$

$$C_{MAX} = max\{C_{\pi^k(n_k),M_2}\}, k = 1, 2, ..., f; \tag{5}$$

3 Proposed Differential Evolution-based Approach

Differential Evolution (DE) [9] is a relatively simple evolutionary algorithm, primarily used to solve continuous optimisation problems. Through the nature-inspired concepts of selection, mutation and crossover, DE iteratively attempts to improve a set of candidate solutions to a particular problem, where a solution is represented by a vector of real values, by updating the population when better solutions are found until some termination criteria is met. DE has previously been used to solve a variety of optimisation problems, with a wide range of DE variants existing in the literature [1].

The general DE framework used in this paper is as follows. The first step is to initialise N solutions $x_1 ... x_N$, where N is the population size, with random values in each dimension. The algorithm then repeats the following steps until some termination criterion is met. At generation G, for each individual $x_{i,G}$ in the population, depending on the mutation strategy used, up to five solutions are chosen at random to use for mutation. The random solutions are used to calculate difference vectors to diversify the solution. A new mutant vector $v_{i,G}$ is generated using the given mutation strategy. In general a DE mutation strategy is referred to in the format: *mutation strategy/number of difference vectors*. Given x_{best} as the best solution found so far, randomly selected solutions x_{r1} to x_{r5} and $F \in [0, 2]$ as a weighting (the differential weight) used to control the influence of different vectors within the mutation, the five mutation strategies used in this paper are as follows:

$$rand/1 \qquad v_{i,G} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) \tag{6}$$

$$rand/2 \qquad v_{i,G} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) + F \cdot (x_{r4,G} - x_{r5,G}) \tag{7}$$

$$best/1 \qquad v_{i,G} = x_{best,G} + F \cdot (x_{r1,G} - x_{r2,G}) \tag{8}$$

$$best/2 \qquad v_{i,G} = x_{best,G} + F \cdot (x_{r2,G} - x_{r3,G}) + F \cdot (x_{r4,G} - x_{r5,G}) \tag{9}$$

$$current - to - best/1 \quad v_{i,G} = x_{i,G} + F \cdot (x_{best,G} - x_{i,G}) + F \cdot (x_{r1,G} - x_{r2,G}) \tag{10}$$

After a mutant vector $v_{i,G}$ has been obtained, crossover is performed between the original target vector $x_{i,G}$ and the mutant vector $v_{i,G}$. In each case we use binomial crossover to generate a trial vector $u_{i,G}$, with the value assigned to each dimension j of the trial vector as follows:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if rand[0,1)} \leq CR \text{ or } j = j_{rand} \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{11}$$

Here $CR \in [0, 1]$ is the crossover probability, controlling which of the two parents, either the target vector or the mutant vector, has the greatest influence on the trial vector generated. j_{rand} is the value of a random dimension to ensure that at least one dimension is taken from each of the parent solutions. Finally the target vector and the trial vector are compared using a fitness function g, with the best of the two kept in the population for the following generation:

$$x_{i,G+1} = \begin{cases} u_{i,G} & \text{if } g(u_{i,G}) < g(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \tag{12}$$

3.1 Encoding and Decoding Solutions to the DTMFSP

As mentioned above, given n jobs, the search process when solving the DTMFSP is over a vector of integers $S = s_1, ..., s_n$, with each $s \in S$ able to take a value between 1 and f corresponding to the factory that job is allocated to. As DE operates over a space of real values, a method to map a DE vector to a DTMFSP solution in discrete space is needed. Assuming that there are n jobs and f factories in the DTMFSP problem, each vector x has n dimensions with values in the range [0,1]. To interpret a vector as a solution to the DTMFSP problem, the range [0,1] is partitioned into n equal parts, that is [0, 1/f), [1/f, 2/f), ..., [(n-1)/f, n). For example with 4 factories and 5 jobs the vector [0.21, 0.85, 0.42, 0.63, 0.31] indicates that the 1st job is assigned to 1st factory (as it is in the interval [0, 0.25)), the 2nd job is assigned to 4th factory (it is within the interval [0.75, 1)), with the remaining jobs assigned to factory 2, factory 3, and factory 2 respectively. After determining which jobs are assigned to which factory based on the DE vector, we can then determine the job sequence of each factory. For each factory, finding the minimum makespan is the classical two machine flow-shop scheduling problem, so we can use Johnson‘s rule to compute the optimal job sequence. The maximum makespan of this schedule can then be computed.

4 Experimental Framework and Parameter Tuning

In our experiments we use the DTMFSP benchmark instances based on a real-world scenario introduced by Deng et al. [2]. This set consists of 100 instances

with 5 instances for each $f \in \{2, 3, 4, 5, 6\}$ and $n \in \{20, 50, 100, 200\}$. Here we will focus on the largest of these instances where $f = 3$, 4 and 5. As Deng et al. [2] used 0.1*n seconds CPU time as a termination criterion on different hardware, only an indirect comparison can be made to their results. For the benefit of future work in this area, we use the number of fitness evaluations as a termination criterion, allowing each run to evaluate 500,000 individuals. All experiments were performed on an Intel Core i7 CPU @ 2.40GHz with 16GB RAM.

The set of possible parameter settings for each DE component are given in Table 1. In order to test all of these parameter settings, we would have to test 625 (5^4) combinations. However, using an orthogonal array we are able to reduce this to 25 combinations, and use the results of these 25 experiments to derive the best set of parameters to use on the full benchmark set. Consistent with the parameter tuning performed by Deng et al. [2], we use the 11th 4-factory instance (F4_11), which has n= 100 jobs to schedule for parameter tuning.

Table 1. Set of possible settings for each DE parameter

Parameter	Possible Values
F	{0.25, 0.50, 0.75, 1.00, 1.25}
CR	{0.02, 0.04, 0.06, 0.08, 0.10}
N	{10, 25, 50, 100, 200}
Mutation strategy	{rand/1, rand/2, best/1, best/2, current-to-best/1}

After applying each of the 25 parameter combinations to F4_11, we are able to define the best set of parameters as DE_{Best}. It was found that the parameter set where $F = 0.50$, $CR = 0.02$, $N = 25$ and mutation strategy = rand/2 performed best, and will be referred to as DE_{Best} herein.

5 Results

Deng et al. [2] presented three Harmony Search variants applied to the DTMFSP: classic Harmony Search (HS), 'improved' Harmony Search (IHS) and Global-best Harmony Search (GHS). Tables 2-4 show the results of the best value obtained from 10 runs of DE_{Best} compared to the results of IHS, HS and GHS obtained by Deng et al. [2] for instances with 4, 5 and 6 factories respectively. Although only an indirect comparison can be made due to the differing termination criteria used, DE_{Best} does not run for a significantly different amount of time to the methods of Deng et al. [2], even when differences in hardware are taken into consideration. The best results obtained for each instance are highlighted bold, with new best results also highlighted with an asterisk(*). Due to space limitations, in the case of 4 factories in Tabale 2, we have only included those instances for which new best results are obtained. For the remaining instances of this type, DE_{Best} and IHS [2] obtain identical results.

Table 2. Best makespan obtained by DE_{Best} and Harmony Search variants over 10 runs of each instance of the Deng et al. [2] benchmark set with $f = 4$ factories

Instance	DE_{Best}	IHS	HS	GHS
F4_4	**312***	313	313	313
F4_11	**1199***	1200	1201	1200
Average (F4_1 - F4_20)	**1171.0**	1171.1	1173.1	1173.0

For the instances with $f = 4$ factories Table 2, DE_{Best} and IHS clearly outperform HS and GHS, with DE_{Best} obtaining new best solutions for instances F4_4 and F4_11. DE_{Best} also outperforms IHS on average in these instances. GHS performs particularly badly on these instances, only obtaining the same best result as the other 3 methods in 1 of the 20 instances, with HS only performing marginally better. As noted by Deng et al. [2], as the number of factories increases, the task of assigning jobs to factories assignment becomes more complex, making it more difficult to find the best combination of jobs for a particular factory. As we see in Table 3 where $f = 5$, HS and GHS are no longer able to find same best result as DE_{Best} for any of these instances. DE_{Best} is the best performing method on 17 of these 20 instances and overall on average, finding new best solutions for 5 instances. Table 4 presents the results when the number of factories $f = 6$. In this set of large instances the performance of DE_{Best} drops off somewhat, although it is able to find new best results for 5 of the 20 instances and obtains the best solution of all methods for 10 of the 20 instances. It is notable here that IHS is the best performing method when the instances get bigger, particularly for instances F6_11-F6_20 where the number of jobs is 100 and 200. One possible reason that DE_{Best} is being outperformed on these larger instances is the nature of the parameter tuning experiments performed. As the parameter tuning was done on slightly smaller instances, where $f = 4$, it could be the case that the parameters of DE_{Best} are overfitted to instances of this size and are not scaling well to the largest instances in the benchmark set. Despite this, DE_{Best} outperforms the three Harmony Search variants presented by Deng et al. [2] on average for instances with $f = 4$ and 5, providing new best-known results for 12 of the 60 benchmark instances tested.

6 Conclusions and Future Work

In this paper we have presented experiments applying Differential Evolution (DE) to the distributed two machine flow-shop scheduling problem (DTMFSP). Although the search space of the problem is discrete, a mapping is defined from continuous to discrete space in order to apply DE to the problem indirectly. An initial set of parameter tuning experiments were performed to decide the parameters for DE before the best combination was applied to a set of benchmark instances and compared to 3 existing methods based on the Harmony Search algorithm. DE was able to outperform the Harmony Search methods in 2 of 3

Table 3. Best makespan obtained by DE_{Best} and Harmony Search variants over 10 runs of each instance of the Deng et al. [2] benchmark set with $f = 5$ factories

Instance	DE_{Best}	IHS	HS	GHS
F5_1	**227**	**227**	229	231
F5_2	**234**	**234**	236	235
F5_3	**243***	244	244	244
F5_4	**272**	**272**	275	275
F5_5	**234**	**234**	236	238
F5_6	**543***	545	547	546
F5_7	**537**	**537**	547	544
F5_8	**518***	519	525	522
F5_9	**534**	**534**	542	542
F5_10	466	**465**	473	469
F5_11	1008	**1007**	1024	1029
F5_12	**1044**	**1044**	1057	1060
F5_13	**982**	**982**	988	985
F5_14	**986***	988	1004	1005
F5_15	**1090***	1092	1096	1097
F5_16	**2079**	**2079**	2086	2089
F5_17	**2131**	**2131**	2140	2143
F5_18	**2087**	**2087**	2094	2095
F5_19	**2050**	**2050**	2073	2069
F5_20	2018	**2017**	2029	2029
Average	**964.2**	964.4	972.3	972.4

sets of instances when clustered by number of factories and provide new best-known results for 12 instances of the 60 tested.

All of the DE methods tested use fixed parameters throughout a run, with the same set of parameters used across all instances of the benchmark set. It is possible that good parameter settings are dependent on the particular instance under consideration, or even the current state of the search. The literature contains existing variations of DE such as SHADE [10] and JADE [12], which control parameters including crossover probability and differential rate adaptively. Future work will apply methods such as these to the DTMFSP to see if any gain can be made by controlling parameters in a dynamic manner.

References

1. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1), 4–31 (2011)
2. Deng, J., Wang, L., Shen, J., Zheng, X.: An improved harmony search algorithm for the distributed two machine flow-shop scheduling problem. In: Harmony Search Algorithm, pp. 97–108. Springer (2016)
3. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of operations research 1(2), 117–129 (1976)
4. Johnson, S.M.: Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly 1(1), 61–68 (1954)

Table 4. Best makespan obtained by DE_{Best} and Harmony Search variants over 10 runs of each instance of the Deng et al. [2] benchmark set with $f = 6$ factories

Instance	DE_{Best}	IHS	HS	GHS
F6_1	**173**	**173**	**173**	174
F6_2	**235**	**235**	**235**	**235**
F6_3	**192***	194	194	194
F6_4	**217***	218	219	219
F6_5	233	**232**	234	236
F6_6	**434***	438	449	443
F6_7	**423***	426	433	428
F6_8	393	**391**	406	405
F6_9	445	**442**	462	459
F6_10	460	**459**	474	465
F6_11	907	**904**	925	926
F6_12	**787**	**787**	798	796
F6_13	**867**	**867**	880	882
F6_14	877	**874**	901	900
F6_15	883	**878**	915	904
F6_16	**1782***	1786	1801	1799
F6_17	1725	**1724**	1740	1739
F6_18	1664	**1662**	1698	1689
F6_19	**1707**	**1707**	1724	1731
F6_20	1680	**1677**	1728	1715
Average	804.2	**803.7**	819.5	817.0

5. Lin, B.M., Hwang, F., Gupta, J.N.: Two-machine flowshop scheduling with three-operation jobs subject to a fixed job sequence. Journal of Scheduling To appear, 1–10 (2017)
6. Onwubolu, G., Davendra, D.: Scheduling flow shops using differential evolution algorithm. European Journal of Operational Research 171(2), 674–692 (2006)
7. Pan, Q.K., Wang, L., Gao, L., Li, W.: An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers. Information Sciences 181(3), 668–685 (2011)
8. Qian, B., Wang, L., Huang, D.x., Wang, W.l., Wang, X.: An effective hybrid de-based algorithm for multi-objective flow shop scheduling with limited buffers. Computers & Operations Research 36(1), 209–233 (2009)
9. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4), 341–359 (1997)
10. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2013). pp. 71–78. IEEE (2013)
11. Wang, L., Pan, Q.K., Suganthan, P.N., Wang, W.H., Wang, Y.M.: A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Computers & Operations Research 37(3), 509–520 (2010)
12. Zhang, J., Sanderson, A.C.: Jade: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13(5), 945–958 (2009)